Slide 7, 8 dari materi Pengantar Sistem Operasi

Definisi Sistem Operasi

- Resource allocator
  - mengatur resource
  - mengalokasikan dan mengontrol pemakaian resources dari berbagai program/aplikasi.
- Control program
  - Mengendalikan eksekusi user program dan pemakaian sistem resource (contoh : operasi pada I/O device) => handal, reliable, terlindung.
- Kernel
  - Sistem program yang berjalan (“ada) terus menerus selama komputer aktif`.
  - Kontras dengan aplikasi yang di “load”, eksekusi dan terminasi .

Fungsi Sistem Operasi

- *Resource Allocator* / *Resource Manager,* mengelola sumber daya sistem komputer secara efisien
  - Mengatur penjadwalan sumber daya (setiap program mendapatkan waktu dan ruang terhadap sumber daya)
- *Control Program* / Program Pengendali
  - Mengatur eksekusi aplikasi dan operasi dari alat I/O untuk mencegah terjadinya kesalahan dan penggunaan yang tidak semestinya
- *Extended Machine* / *Virtual Machine,* menyediakan sekumpulan layanan yang disebut *system call*
  - Menyembunyikan kerumitan pemrograman hardware
  - Sebagai landasan/basis untuk program aplikasi lain

2. Slide 3 dari materi Struktur Sistem Operasi

Manajemen Proses

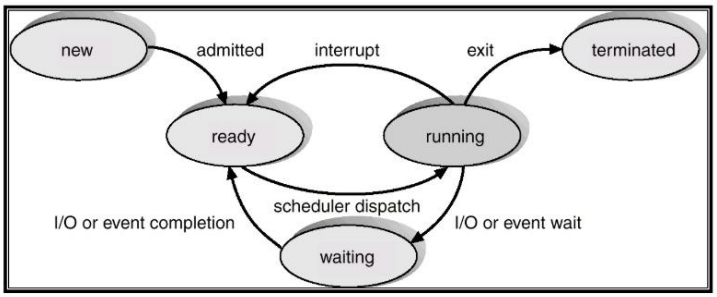
- Proses adalah sebuah program yang sedang dijalankan (eksekusi).
  - Suatu proses memerlukan resources pada saat ekesekusi: CPU time, memory, files dan I/O devices
- Sistim operasi bertanggung jawab terhadap aktifitas yang berhubungan dengan manajemen proses:
  - Process creation & deletion.
  - Process suspension (block) & resumption.
  - Mekanisme:
    - Sinkronisasi antar proses
    - Komunikasi antar proses

3. Slide 4, 5, 13, 16 dari materi Manajemen Proses

Status Proses

- Saat-saat proses dijalankan (executed) maka status dari proses akan berubah
  - Status proses tidak selamanya aktif menggunakan CPU).
  - Sering proses menunggu I/O complete => status wait, sebaiknya CPU diberikan kepada proses yang lain.
  - Mendukung multi-tasking – utilisasi CPU dan I/O
- Status proses (antara lain):
  - **new**: proses dibuat.
  - **running**: instruksi dieksekusi.
  - **waiting**: proses menunggu beberapa event yang akan terjadi
  - **ready**: proses menunggu jatah waktu dari prosessor
  - **terminated**: proses selesai dieksekusi.

Diagram Status Proses



Penjadual / Schedulers

- Bagaimana schedulers memilih proses atau program (decision)?
  - Lebih dari satu proses atau program yang akan dijalankan?
- Long-term scheduler (or job scheduler) – memilih proses/program yang mana yang akan di load dan berada di **ready queue**.
  - Kemungkinan terdapat proses atau job baru.
  - Kemungkinan proses dipindahkan dari memori ke disk (swap out).
- Short-term scheduler (or CPU scheduler) – memilih proses yang mana yang berada di **ready queue** akan “run” (mendapatkan jatah CPU).

Alih Konteks / Context Switch

- Jika Scheduler switch ke proses lain, maka sistim harus menyimpan “informasi” proses sekarang (supaya dapat dijalankan kembali)
- Load “informasi” dari proses baru yang berada di PCB
- Waktu Context-switch adalah overhead; sistem tidak melakukan pekerjaan saat terjadi switch.
  - Sangat tergantung pada waktu di hardware
  - OS modern mencari solusi untuk mengurangi overhead waktu switch proses

4. Slide 6, 7, 9, 12, 22 dari materi Penjadwalan CPU

Penjadualan CPU

- Keputusan penjadwalan CPU mempertimbangkan 4 keadaan sbb :
  1. Ketika proses berpindah dari state running ke state waiting (contoh: permintaan I/O, atau menunggu terminasi dari satu proses child)
  2. Ketika proses berpindah dari state running ke state ready (contoh: saat terjadi interrupt)
  3. Ketika proses berpindah dari state waiting ke state ready (contoh: menyelesaikan I/O)
  4. Ketika proses diterminasi
- Penjadualan 1 dan 4 termasuk nonpreemptive
- Penjadualan lainnya termasuk preemptive

Jenis Penjadualan

- Non-preemptive: setiap proses secara sukarela (berkala) memberikan CPU ke OS.
- Preemptive: OS dapat mengambil (secara interrupt, preempt) CPU dari satu proses setiap saat.
  - *Prasyarat untuk OS real-time system*

Kriteria Penjadualan

- Utilisasi CPU
  - menjadikan CPU terus menerus sibuk (menggunakan CPU semaksimal mungkin).
- Throughput
  - jumlah proses yang selesai dijalankan (per satuan waktu).
- Turn around time
  - waktu selesai eksekusi suatu proses (sejak di submit sampai selesai).
- Waiting time
  - waktu tunggu proses (jumlah waktu yang dihabiskan menunggu di ready queue).
- Response time
  - waktu response dari sistim terhadap user (interaktif, time-sharing system), sehingga interaksi dapat berlangsung dengan cepat.

First Come First Served (FCFS)

- Algoritma:
  - Proses yang request CPU pertama kali akan mendapatkan jatah CPU.
  - Sederhana – algoritma maupun struktur data: menggunakan FIFO queue (ready queue).
- Termasuk non preemptive
  - Timbul masalah “waiting time” terlalu lama jikadidahului oleh proses yang waktu selesainya lama.
    - Tidak cocok untuk time-sharing systems.
    - Digunakan pada OS dengan orientasi batch job.

Round Robin (1)

- Konsep dasar : *time sharing*
- Sama dengan FCFS yang bersifat preemptive
- Quantum time untuk membatasi waktu proses
- Jika CPU burst < *Quantum time*, proses melepaskan CPU jika selesai dan CPU digunakan untuk proses selanjutnya
- Jika CPU burst > *Quantum time*, proses dihentikan sementara dan mengantri di ekor dari *ready queue*, CPU menjalankan proses berikutnya

First Come First Served (FCFS)

- Algoritma:
  - Proses yang request CPU pertama kali akan mendapatkan jatah CPU.
  - Sederhana – algoritma maupun struktur data: menggunakan FIFO queue (ready queue).
- Termasuk non preemptive
  - Timbul masalah “waiting time” terlalu lama jikadidahului oleh proses yang waktu selesainya lama.
    - Tidak cocok untuk time-sharing systems.
    - Digunakan pada OS dengan orientasi batch job.

Contoh 1

Proses	Burst Time
P_1	24
P_2	3
P_3	3

- Diketahui proses yang tiba adalah P_1, P_2, P_3. Gant chart-nya adalah :

- Waiting time untuk *P1* = 0; *P2* = 24; *P3* = 27
- Average waiting time: (0 + 24 + 27)/3 = 17

Contoh 2

- Diketahui proses yang tiba adalah P_2, P_3, P_1. Gant chart-nya adalah :

- Waiting time untuk *P1* = 6*; P2* = 0*; P3* = 3
- Average waiting time: (6 + 0 + 3)/3 = 3
  - Lebih baik dari kasus sebelumnya
- *Convoy effect* proses yang panjang diikuti proses yang pendek

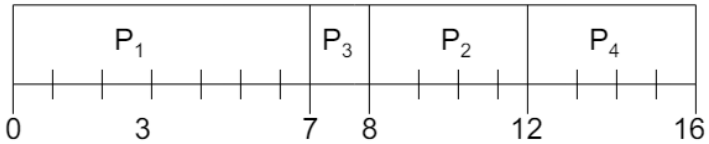
Shortest-Job-First (SJF)

- Proses yang memiliki CPU burst paling kecil dilayani terlebih dahulu.
- Terdapat 2 skema :
  - *nonpreemptive* – CPU hanya satu kali diberikan pada suatu proses, maka proses tersebut tetap akan memakai CPU hingga proses tersebut melepaskannya
  - *preemptive* –jika sisa waktu proses pertama lebih besar dari proses kedua, maka proses pertama dihentikan dan diganti proses kedua. (dikenal dengan Shortest-Remaining-Time-First /SRTF).
- SJF akan optimal, ketika rata-rata waktu tunggu minimum untuk set proses yang diberikan

Contoh Non-Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (non-preemptive)

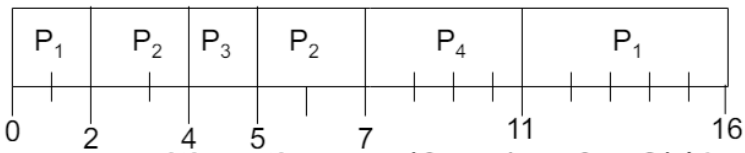


- Average waiting time = (0 + 6 + 3 + 7)/4 = 4

Contoh Preemptive SJF

Process	Arrival Time	Burst Time
P_1	0.0	7
P_2	2.0	4
P_3	4.0	1
P_4	5.0	4

- SJF (preemptive)



- Average waiting time = (9 + 1 + 0 +2)/4 = 3

Contoh lain

Proses	Arrival Time	Burst Time
P1	0	6
P2	2	8
P3	3	7
P4	5	3

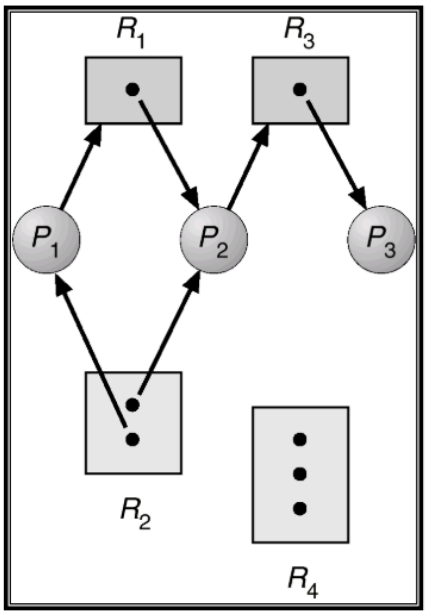
Shortest Job First

Proses	Arrival Time	Burst Time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

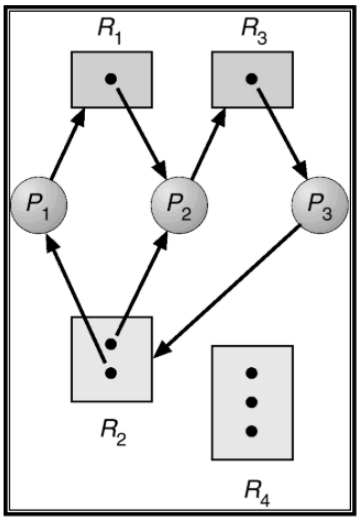
Shortest remaining time first

7. Deadlock, pahami cara menggambar resources allocation graph, dan syarat terjadinya deadlock.

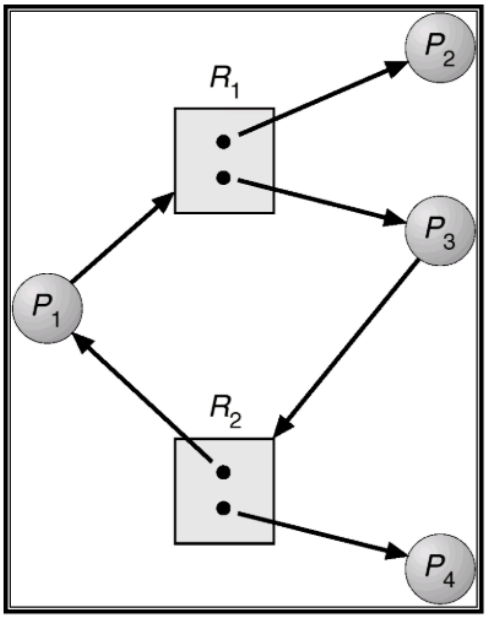
Contoh Resource Allocation Graph



Graf Resource Allocation Dengan Deadlock



Graf Resource Allocation dengan Cycle Tanpa Deadlock



Kondisi yang Diperlukan untuk Terjadinya Deadlock

- **Mutual Exclusion**
  - Serially-shareable resources (mis. Buffer)
  - Contoh: Critical section mengharuskan mutual exclusion (termasuk resource), sehingga potensi proses akan saling menunggu (blocked).
- **Hold & wait :**
  - Situasi dimana suatu proses sedang hold suatu resource secara eksklusif dan ia menunggu mendapatkan resource lain (wait).

Kondisi yang Diperlukan untuk Terjadinya Deadlock (cont.)

- **No-Preemption Resource :**
  - Resource yang hanya dapat dibebaskan secara sukarela oleh proses yang telah mendapatkannya
  - Proses tidak dapat dipaksa (pre-empt) untuk melepaskan resource yang sedang di hold
- **Circular wait**
  - Situasi dimana terjadi saling menunggu antara beberapa proses sehingga membentuk waiting chain (circular)
  - Misalkan proses (P0, P1, .. Pn) sedang blok menunggu resources: P0 menunggu P1, P1 menunggu P2, .. dan Pn menunggu P0.

Deadlock Prevention

- Pencegahan: Faktor-faktor penyebab deadlock yang harus dicegah untuk terjadi
- 4 faktor yang harus dipenuhi untuk terjadi deadlock:
  - Mutual Exclusion: pemakaian resources.
  - Hold and Wait: cara menggunakan resources.
  - No preemption resource: otoritas/hak.
  - Circular wait: kondisi saling menunggu.
- Jika salah satu bisa dicegah maka deadlock pasti tidak terjadi!